Bases de Datos en Visual Basic (1)

Visual Basic nos permite trabajar directamente con distintas bases de datos (ACCESS, dBaseIII, dBaseIV , dBase 5, Excel3, Excel4, Excel5, Excel7, FoxPro2.x, Foxpro3.0, LotusWK1, LotusWK3, LotusWk4, Paradox3.x, Paradox4.x y Paradox5.x

Para acceder a estas Bases de Datos basta con introducir un control Data en el formulario, y fijarle las propiedades apropiadas para que trabaje sobre uno u otro tipo de base de datos. El control Data nos permite acceder de una forma sencilla a cualquier base de datos de estos tipos, y sirve de enlace entre la base de datos y los controles que son habilitados para presentar los datos de esa base.

El Control Data

El control Data puede tomarse directamente de la caja de herramientas. En el formulario tiene el aspecto de una barra deslizante:

El control Data proporciona acceso a datos almacenados en bases de datos usando uno de los tres tipos de objetos **Recordset**. El control Data le permite ir de registro en registro y presentar y manipular los datos de los en controles enlazados. Sin un control Data, los **controles enlazados con datos** de un formulario no pueden tener acceso automáticamente a los datos.

Los **controles enlazados** solamente pueden tener acceso a un control **Data** si este está en el mismo Formulario.

En el tema de Bases de Datos se emplean términos no conocidos aún. Se irán viendo a lo largo del curso, pero no queda otro remedio mas que comenzar a utilizarlos. Se irán haciendo avances de estos términos, que serán explicados en profundidad en su momento.

Avance de los términos empleados:
Objeto Recordset (conjunto de registros)

Es un conjunto lógico de registros. Los tres tipos de objetos Recordset son **dynaset**, (Permite la lectura y escritura de un registro) **snapshot** (Realiza una lectura instantánea de los registros, no permitiendo modificarlos) y **table**. (Representación en el código de una tabla base que puede utilizarse para agregar, modificar o eliminar registros de una sola tabla).

Controles enlazados
Son los controles que pueden presentar directamente datos de uno o varios campos de una Base de Datos. Los controles DBList, DBCombo y DBGrid tienen la posibilidad de presentar un conjunto de registros cuando se asocian con un control Data. Los controles CheckBox, TextBox, Label, Picture, Image, ListBox y ComboBox también son controles enlazados con datos y pueden asociarse a un único campo de un Recordset administrado por un control Data.

La mayoría de las operaciones de acceso a datos se pueden realizar usando el control Data sin escribir ningún código. Los controles enlazados con un control Data presentan de forma automática los datos de uno o más campos del registro actual o, en algunos casos, de un conjunto de registros a ambos lados del **registro actual**. El control Data realiza todas las operaciones sobre el **registro actual**.

Avance de términos
Registro Actual. Un registro es un conjunto completo de campos. Una base

puede tener muchos registros, pero el “puntero” de la base de datos apunta a un único registro en cada momento. Ese registro al que apunta el puntero se llama registro actual.

Si el control Data recibe instrucciones de moverse a un registro diferente, todos los controles enlazados pasan automáticamente los cambios al control Data para ser guardados en la base de datos. El control Data se sitúa después en el registro requerido y pasa los datos del registro actual a los controles enlazados donde son presentados. Esto significa que se pueden modificar los datos de una base de datos simplemente cambiando los datos en los controles enlazados que lo permitan, y moviendo el puntero de la base de datos, es decir, cambiando el registro actual.

Una vez iniciada la aplicación, Visual Basic usa las propiedades del control Data para abrir la base de datos seleccionada, abrir un objeto Database y crear un objeto Recordset. Las propiedades Database y Recordset del control Data hacen referencia a los objetos Database y Recordset recién creados que pueden ser manipulados por el control Data. Siempre podremos conocer el Recordset usado por el control Data leyendo esa propiedad

VariabletipoRecordset =Data1.**Recordset**

Cuando se usa un control Data para crear un objeto Recordset o cuando se crea un objeto Recordset en el código y se asigna al control Data, el motor de base de datos Jet de Microsoft puebla automáticamente el objeto Recordset. Como resultado, los marcadores (y en los objetos Recordset de tipo snapshot, los datos del conjunto de registros) se guardan en la memoria local; el usuario no necesita manipular el control Data y no es necesario invocar el método MoveLast en el código. Los bloqueos de página usados para crear el Recordset se liberan más rápidamente, haciendo posible que otros objetos Recordset accedan a los mismos datos. Los objetos Recordset creados en el código pero que no se asignan a un

control Data no son poblados automáticamente por el motor Jet. Se deben poblar desde el código.

El párrafo anterior exige al menos una explicación.
Cuando se crea un Recordset mediante un control Data, se leen inmediatamente todos los registros que forman parte de ese Recordset (Recuerde que un Recordset es un conjunto de registros). De esta forma, en una base que estuviera compartida por varios usuarios a través de una Red de Area Local (RAL) un usuario leería todos los datos en el mismo momento de la creación del Recordset por el control Data, llevaría esos datos a su memoria RAM y no volvería a estorbar en la base de datos (cuando un usuario de una RAL lee un dato en una BD, bloquea esta BD mientras dura su lectura y no pueden acceder a ella otros usuarios) Si el Recordset se crea por código, se lee solamente un registro (la base se bloquea en el momento de la lectura e inmediatamente se libera), y cuando se le pide otra operación (p.e. que avance un registro) vuelve a bloquear la BD, lee ese registro y la desbloquea. En principio el leer el Recordset de una vez parece que tiene ventajas en aquellas instalaciones que tienen una base de datos compartida. Todo ello a un precio. Ocupar mas memoria RAM en cada uno de los PCs de los usuarios. Esta limitación hace en algún caso que no sea posible utilizar un control Data por falta de memoria RAM en los puestos de usuario.

El control Data puede manipularse con el mouse, moviéndose de registro en registro o al principio o al final del Recordset. El control Data no permite que el usuario se pase de los límites del Recordset usando el mouse. No se puede mover el enfoque al control Data.

Observación muy importante

El control Data crea un objeto Database y un objeto Recordset automáticamente. Estos objetos de acceso a datos son idénticos a los creados mediante código, y tienen las mismas propiedades y métodos. Podemos referirnos a ellos usando el nombre del control Data seguido del nombre del objeto (Database o Recordset). Por ejemplo:

Data1.Database **Data1.Recordset**

El objeto Database creado por un control Data no se cierra aunque se cambie la propiedad DatabaseName del control Data. Lo mismo ocurre con el objeto Recordset. Solamente podemos cerrarlos utilizando el método Close:

Data1.Database.**Close** Data1.Recordset.**Close**

Esta observación debe ser tenida muy en cuenta sobre todo cuando la el control Data abre la Base de Datos de forma exclusiva, o cuando tenemos que hacer una operación con la Base de Datos que exija que esté cerrada. Por ejemplo, el método **CompactDatabase** y otros métodos que veremos más adelante.

Objetos para acceso a datos

Los objetos para acceso a datos **Database** y **Recordset** creados por el control **Data** pueden usarse en los procedimientos. Los objetos **Database** y **Recordset** tienen cada uno sus propiedades y métodos propios y se pueden escribir procedimientos que usen estas propiedades y métodos para manipular los datos.
Por ejemplo, el método **MoveNext** de un objeto Recordset mueve el registro actual al siguiente registro del Recordset. Para invocar este método, se podría usar el siguiente código:

Data1.Recordset.**MoveNext**

El control Data tiene la capacidad de tener acceso a cualquiera de los tres tipos de objetos Recordset del motor Jet versión 2.0. Si no se selecciona el tipo de conjunto de registros, se crea un Recordset de tipo Dynaset.

Nota. Las constantes usadas para requerir un tipo específico de Recordset cuando se usa un control Data son diferentes de las constantes usadas para determinar el tipo de Recordset creado o que se va crear usando el método OpenRecordset.

Para seleccionar un tipo específico de **Recordset**, establezca la propiedad **RecordsetType** del control Data a:

Tipo de Recordset	Valor	Constante control Data	Constante OpenRecordset
Table	0	vbRSTypeTable	dbOpenTable
Dynaset	1	vbRSTypeDynaset	dbOpenDynaset
Snapshot	2	vbRSTypeSnapshot	dbOpenSnapshot

Diferencias entre la Edición Standard y la Edición Profesional de Visual Basic

En cuanto al acceso a datos, la diferencia principal entre las ediciones Estándar y la Profesional de Visual Basic es la capacidad de crear nuevos objetos para acceso a datos. En la edición **Estándar** NO se pueden declarar en el código (con la palabra clave Dim) variables como objetos para acceso a datos. Esto quiere decir que sólo el control Data puede crear objetos Database y Recordset.
En la edición **Profesional** de Visual Basic, Si se puede crear un nuevo objeto Recordset y asignarlo a la propiedad Recordset del control Data. Todos los controles enlazados conectados con el control Data permiten la manipulación de los registros del Recordset creado. Asegúrese de que las propiedades DataField

de los controles enlazados están definidas con nombres de campo válidos del nuevo Recordset.

¡Esta diferencia ha llevado a la locura a muchos alumnos y programadores usando la versión de su casa y la de su empresa o centro escolar!

Consultas almacenadas

Otra opción importante cuando se usa el control Data es la posibilidad de ejecutar consultas almacenadas. Si se ha creado previamente un objeto **QueryDef**, (No se asuste, Un QueryDef no es ni mas ni menos que una consulta en la base de datos ACCES) el control Data puede ejecutarlo y crear un Recordset usando las propiedades almacenadas **SQL**, Connect y otras del objeto QueryDef. Para ejecutar un QueryDef, establezca la propiedad RecordSource del control Data al nombre del objeto QueryDef (En vez de poner el nombre de una tabla, ponga el nombre de la consulta. Cuando está rellenando esta propiedad, le aparecen, al mismo tiempo que el nombre de las tablas, el nombre de las consultas existentes). Si el objeto QueryDef almacenado contiene parámetros, es necesario crear el Recordset y pasárselo al control Data.

Es mucho más rápida una consulta utilizando un QueryDef que introduciendo la consulta SQL. La razón es muy sencilla. Al crear un QueryDef (Ya verá como) el motor Jet (Es decir ACCESS) crea esa consulta en la propia base de datos. Si lo que hace es una consulta SQL, se debe obtener la información registro a registro según las condiciones establecidas en la cláusula SQL. Esta segunda opción tarda lógicamente más.

Administración de BOF/EOF

BOF (Begin Of File). Se produce el BOF cuando el control Data se posiciona sobre el registro inmediatamente anterior al primero (No es un juego de palabras). Este

registro será el -1.

EOF (End Of File) Se produce la condición EOF cuando el control Data se posiciona en el registro inmediatamente posterior al último. Este registro será también el -1.

El control Data también puede controlar la situación en la que se encuentra con objetos Recordset sin registros. Cambiando la propiedad **EOFAction**, se puede programar el control Data para que pase automáticamente al modo **AddNew**.
El control Data puede programarse para que se ajuste automáticamente a la parte superior o inferior de su formulario primario usando la propiedad **Align**. En cualquier caso, el control Data ajusta su tamaño horizontal al de su formulario primario cuando el tamaño de éste cambia. Esta propiedad permite situar un control **Data** en un formulario **MDI** sin requerir un control **Picture** que lo contenga.

Propiedades del control Data

Align El hecho de que tenga esta propiedad nos permite colocar un control **Data** en un formulario MDI hijo.

Appearance Flat y 3-D

Backcolor Color de fondo de la parte intermedia del control

BOFAction Esta propiedad establece el comportamiento del Control Data cuando tiene seleccionado como registro actual el registro -1 (los registros van del 0 a un número positivo). Esta situación se da cuando se va bajando el número del registro actual mediante el método **MovePrevious** y nos pasamos del registro 0, o cuando el Recordset no tiene ningún registro. En esta propiedad podemos establecer que se mueva al primer registro (**MoveFirst**) o que se quede donde está (**BOF** Begin Of File).

Caption El nombre que figurará en la parte intermedia del control.

Connect Muy Importante. En esta propiedad debemos indicarle al control Data el tipo de base de datos a la que va a conectarse. Admite todas las bases enumeradas al principio de este capítulo.

DatabaseName En esta propiedad se le indica el nombre (Con su Path) de la base de datos a la que debe conectarse. Para facilitar la búsqueda de la base de datos, haciendo click en esta propiedad en la caja de propiedades, podemos sacar un cuadro de diálogo haciendo click de nuevo en los tres puntos que aparecen a la derecha de la propiedad. El cuadro de diálogo seleccionará directamente las extensiones de los ficheros de bases de datos acordes con el tipo de base de datos seleccionada en la propiedad **Connect**.

Database (Solo en ejecución)

Esta propiedad es sólo de lectura. Devuelve una referencia a un objeto Database subyacente de un control Data.

Sintaxis Variable = nombredelcontroldata.**Database**

El objeto Database creado por el control Data se basa en las propiedades DatabaseName, Exclusive, ReadOnly y Connect del control.
Los objetos Database tienen propiedades y métodos que puede utilizar para administrar los datos. Puede utilizar cualquier método de un objeto Database con la propiedad Database de un control Data, como Close y Execute. También puede examinar la estructura interna de la Database empleando su colección TableDefs y, a su vez, las colecciones Fields e Indexes de objetos TableDef individuales.

Aunque puede crear un objeto Recordset y pasarlo a la propiedad Recordset

de un control Data, no puede abrir una base de datos y pasar el objeto Database recién creado a la propiedad Database del control Data.

DragIcon, **DragMode** , **Enabled** Igual que todos los controles.

EditMode

Solo lectura. Devuelve un valor que indica el estado de modificación del registro actual.

Sintaxis Variable = objeto.**EditMode**

Variable tomará uno de los siguientes valores:

dbEditNone No se está realizando ninguna operación de modificación.
dbEditInProgress Se ha invocado el método Edit y el registro actual se encuentra en el búfer de copia.
dbEditAdd Se ha invocado el método AddNew y el registro actual del búfer de copia es un registro nuevo que aún no se ha guardado en la base de datos.
La propiedad **EditMode** es especialmente útil cuando se desea partir de la funcionalidad predeterminada de un control Data, o cuando no se utiliza un control Data en Visual Basic Edición profesional. Puede comprobar el valor de la propiedad **EditMode** y el del parámetro acción del procedimiento del evento Validate para determinar si se debe invocar el método Update.
También puede comprobar si el valor de la propiedad **LockEdits** es True y el de **EditMode** es dbEditInProgress para determinar si la página de datos actual se encuentra bloqueada.

EOFAction Establece el comportamiento del control **Data** cuando se sobrepasa

el último registro. Podemos indicarle que se mueva al último registro (**MoveLast**), que se quede donde está (**EOF**), o que introduzca un nuevo registro (**AddNew**)

Exclusive Devuelve o establece un valor (True / False) que indica si la base de datos está abierta para acceso de un único usuario o de múltiples usuarios.

Font, **ForeColor**, **Height**, **Index**, **Left**, **MouseIcon**, **MousePointer**, **Name**, igual que el resto de los controles.

Negotiate Propiedad característica de los controles que tienen la propiedad **Align**. Establece un valor que determina si se muestra un control que puede alinearse cuando un objeto activo del formulario muestra una o más barras de herramientas. No está disponible en tiempo de ejecución.

Options Devuelve o establece un valor que especifica una o más características del objeto Recordset de la propiedad Recordset del control Data. Puede ponerse en el cuadro de propiedades o en tiempo de ejecución. Puede tomar los siguientes valores o nombres de constantes (estos nombres solo si los introduce en tiempo de ejecución)

1 dbDenyWrite En un entorno multiusuario, otros usuarios no pueden realizar cambios en registros del Recordset.

2 dbDenyRead En un entorno multiusuario, otros usuarios no pueden leer registros (sólo Recordset de tipo tabla).

4 dbReadOnly No se pueden realizar cambios en registros del Recordset.

8 dbAppendOnly Puede agregar nuevos registros al Recordset, pero no puede leer los registros existentes.

16 dbInconsistent Las actualizaciones pueden aplicarse a todos los campos del Recordset, aunque infrinjan la condición de unión.

32 dbConsistent (Predeterminado) Las actualizaciones sólo se aplican a los campos que no infringen la condición de unión.

64 dbSQLPassThrough Cuando se utilizan controles Data con una instrucción SQL en la propiedad RecordSource, envía la instrucción SQL a una base de datos ODBC, como SQL Server o Oracle, para su procesamiento.

256 dbForwardOnly El Recordset es un desplazamiento sólo hacia adelante. El único método de movimiento permitido es MoveNext. Esta opción no puede utilizarse en objetos Recordset manipulados con el control Data.

512 dbSeeChanges Genera un error interceptable si otro usuario está cambiando datos que usted edita.

Para establecer más de un valor de los descritos, basta con sumar sus valores. También puede establecer más de un valor para esta propiedad, combinando opciones sumando valores entre sí. Por ejemplo, para establecer **dbReadOnly** y **dbInconsistent** puede utilizar este código:

```
Data1.Options = dbAppendOnly + dbInconsistent
```

Para determinar si la propiedad contiene un valor específico, puede utilizar el operador And. Por ejemplo, para averiguar si el Recordset está abierto para acceso de sólo lectura, podría usar este código:

```
If Data1.Options And dbReadOnly Then...
```

Si cambia la propiedad **Options** en tiempo de ejecución, deberá utilizar el método **Refresh** para que el cambio sea efectivo.

ReadOnly Devuelve o establece un valor que determina si la Database del control

está abierta para acceso de sólo lectura.

RecordsetType

Devuelve o establece un valor que indica el tipo de objeto Recordset que desea que cree el control Data.

Los valores o nombre de la constante que puede adoptar son los siguientes:

0 **vbRSTypeTable** Un Recordset de tipo tabla.
1 **vbRSTypeDynaset**(Predeterminado) Un Recordset de tipo hoja de respuestas dinámica.
2 **vbRSTypeSnapshot** Un Recordset de tipo instantánea.

Si no especifica un **RecordsetType** antes de que el control Data cree el Recordset, se creará un Recordset de tipo hoja de respuestas dinámica. (**Dynaset**)

Recordset

Devuelve o establece un objeto Recordset definido por las propiedades de un control Data o por un objeto Recordset existente.

Veremos mas adelante las propiedades de un objeto Recordset, propiedades que son en todo aplicables al Recordset del control Data.

El Recordset es como se adelantó, un conjunto de registros. Si las propiedades Connect, DatabaseName, Options, RecordSource, Exclusive, ReadOnly y RecordsetType establecidas para el control Data son válidas, se crea un Recordset automáticamente basándose en dichas propiedades. Ese será el

Recordset del control Data. Pero también puede crearse previamente un Recordset mediante la instrucción OpenRecordset y forzar que el Recordset del control Data sea justamente ese mediante la instrucción **Set Data1.Recordset = MiRecordset**

Sea cual fuera la forma de crearlo, a partir del momento en que el Data tenga su Recordset podemos referirnos a él, por ejemplo para ir al primer registro (Data1.Recordset.MoveFirst), avanzar un registro (Data1.Recordset.MoveNext), al anterior (Data1.Recordset.MovePrevious) ir al último (Data1.Recordset.MoveLast), añadir un registro (Data1.Recordset.AddNew), guardar los cambios en la Base de Datos (Data1.Recordset.Update), o borrar el registro actual (Data1.Recordset.Delete)

Si se cambia alguna de las propiedades citadas al principio que cambien el Recordset, es necesario volver a crearlo. Para ello basta con utilizar el Método Refresh. (Data1.Refresh)

RecordSource

Devuelve o establece la tabla, el objeto QueryDef (Consulta) o la instrucción SQL subyacente para un control Data. Esta propiedad puede fijarse en el cuadro de propiedades del control data, (el caso más usado) o introducirse como código. En los dos primeros casos, lo normal es introducir el nombre de una tabla o una consulta de las existentes en la base de datos especificada en la propiedad DatabaseName, nombres que se pueden elegir desplegando la lista que se obtiene haciendo click sobre la flecha que aparece al lado de la casilla de esta propiedad. Observe que en la lista desplegada figuran los nombres de las tablas y de las consultas que tiene la Base de Datos elegida en la propiedad DatabaseName. Ni que decir tiene que para poder introducirla de esta forma es necesario fijar previamente la propiedad DatabaseName.

Puede introducirse en tiempo de ejecución mediante código con la siguiente expresión:

objeto.**RecordSource** [= valor]

Donde objeto es el nombre del control Data, y valor es una expresión de cadena que especifica el nombre de una **Tabla** o una **Consulta**, de las que componen la base de datos especificada en la propiedad DatabaseName, o una consulta SQL válida que utiliza sintaxis apropiada para la base de datos especificada en la propiedad DatabaseName, o una QueryDef, que es una consulta almacenada, que como en los casos anteriores, debe referirse a la base de datos especificada en la propiedad DatabaseName.

La propiedad **RecordSource** especifica el origen de los recursos accesibles a través de controles enlazados del formulario. Si establece la propiedad **RecordSource** como el nombre de una tabla existente en la base de datos, todos los campos de esa tabla serán visibles a los controles enlazados adjuntos a este control Data. El orden de los registros recuperados lo establece el objeto Index que selecciona mediante la propiedad Index del Recordset. Si no establece la propiedad Index, los datos se devolverán sin ningún orden concreto.

Si establece la propiedad **RecordSource** como el nombre de una consulta (QueryDef) existente en la base de datos, todos los campos devueltos por la QueryDef serán visibles a los controles enlazados adjuntos al control Data. El orden de los registros recuperados lo establece la consulta del objeto QueryDef. Si la QueryDef no especifica un orden, los datos se devolverán sin ningún orden concreto.

Si establece la propiedad **RecordSource** como una instrucción SQL que devuelve registros, todos los campos devueltos por la consulta a SQL serán visibles a los controles enlazados adjuntos al control Data. Esta instrucción puede incluir una

cláusula ORDER BY para cambiar el orden de los registros devueltos por el Recordset creado por el control Data o una cláusula WHERE para filtrar los registros.

Después de cambiar el valor de la propiedad **RecordSource** en tiempo de ejecución, deberá utilizar el método **Refresh** para activar el cambio.

Nota. Asegúrese de que cada control enlazado tiene un valor válido para su propiedad **DataField**. Si cambia el valor de la propiedad **RecordSource** de un control Data y, a continuación, utiliza **Refresh**, el Recordset identificará el nuevo objeto. Esto puede invalidar los valores de **DataField** de controles enlazados y producir un error interceptable.

Las Propiedades **Tag**, **Top**, **Visible**, **WhatThisHelpID**, **Width**, igual que el resto de los controles.

METODOS DEL CONTROL DATA

(Se explican aquí los métodos que inciden directamente en el tratamiento de bases de datos. Obviamos los métodos Drag, Move y ZOrder que son idénticos a los del resto de controles)

Método Refresh

De momento es aplicable al control Data. Veremos que también es aplicable a otros objetos de acceso a datos (QueryDef). **El método Refresh no puede utilizarse con colecciones que no sean persistentes, como Databases, Recordsets o Workspaces**.

Actualiza los objetos de una colección para reflejar la base de datos actual. El

método Refresh también se utiliza para cerrar y volver a generar el objeto Recordset o las estructuras de datos creadas por un control Data.

Sintaxis nombredelcontrolData.**Refresh**

Debe utilizar el método **Refresh** en los entornos multiusuario en los que otros usuarios puedan modificar la base de datos. También puede ser necesario su uso en las colecciones afectadas de forma indirecta por los cambios en la base de datos.

Puede utilizar el método **Refresh** sobre un control Data para abrir o reabrir la base de datos (si han variado las propiedades DatabaseName, ReadOnly, Exclusive o Connect) y volver a generar el objeto Recordset indicado por la propiedad Recordset del control.

Método **UpdateControls**

Obtiene el registro actual del objeto Recordset de un control Data y muestra los datos de los controles vinculados al control Data.

Sintaxis nombredelcontroldata.**UpdateControls**

Utilice este método para restablecer en los controles enlazados sus valores originales, por ejemplo cuando un usuario modifica los datos y luego decide cancelar los cambios.
Este método produce el mismo efecto que hacer actual de nuevo al registro actual, excepto en que no se produce ningún evento ni introduce en la Base de Datos los posibles valores que se hubiesen cambiado en los controles enlazados.

Método **UpdateRecord**

Guarda los valores actuales de los controles enlazados.

Sintaxis nombredelcontrolData.**UpdateRecord**

Puede utilizar este método para guardar el contenido actual de los controles enlazados en la base de datos durante el evento Validate sin desencadenar de nuevo el evento Validate. El uso de este método evita los eventos en cascada.
El método **UpdateRecord** tiene el mismo efecto que ejecutar el método **Edit**, modificar un campo y luego ejecutar el método **Update**, excepto en que no se produce ningún evento.
Puede utilizar este método para evitar desencadenar el evento Validate.

En algunos casos, la actualización puede no tener lugar, debido a que la operación vulnere las restricciones de integridad referencial, o que la página que contiene el registro esté bloqueada, o que la base de datos u objeto Recordset no sean actualizables, o a que el usuario no cuente con el permiso adecuado para la operación. En cualquiera de estas circunstancias, se producirá un error interceptable.

Estos son los métodos del Control Data. Este control tiene su Recordset, y el Recordset del control Data tiene sus métodos, idénticos a los de un Recordset creado por código.

EVENTOS DEL CONTROL DATA

Error

Se produce solamente como resultado de un error de acceso a datos que tiene lugar cuando no está ejecutando código Visual Basic.

Private Sub Data1_Error ([índice As Integer,] datoserr As Integer, respuesta As Integer)

índice es el índice de ese control si existen varios con el mismo nombre
datoserr es el número del error.
Respuesta es un número que se corresponde con la respuesta que se va a tomar, como se describe a continuación:

Constante	Valor	Descripción
vbDataErrContinue	0	Continuar.
vbDataErrDisplay	1	(Predeterminado) Presentar el mensaje de error.

Normalmente las aplicaciones proporcionan la funcionalidad del control de los errores de tiempo de ejecución dentro del código. Sin embargo, los errores de tiempo de ejecución pueden ocurrir cuando no se está ejecutando código de la aplicación, por ejemplo cuando:

Un usuario hace clic en un botón del control Data.
El control Data abre automáticamente una base de datos y carga un objeto Recordset después de un evento Form_Load.
Un control personalizado ejecuta una operación como los métodos MoveNext, AddNew o Delete.

Si se produce un error en cualquiera de estas acciones, se produce el evento **Error**.
Si no se escribe código para el procedimiento de evento **Error**, Visual Basic presenta el mensaje asociado con el error.

Reposition

Se produce después de que un registro se convierte en el registro actual.

Private Sub Objeto_**Reposition**()

Donde **objeto** = Nombre del control Data
Cuando se carga un control Data, El primer registro de su objeto Recordset se convierte en el registro actual, provocando el evento **Reposition**. Cuando quiera que un usuario haga clic en uno de los botones del control Data, moviéndose de registro en registro o si se usa uno de los métodos Move del objeto Recordset asociado al control data, como MoveNext, MoveFirst, MovePrevious, los métodos Find, como FindFirst, FindNext, o cualquier otra propiedad o método que cambie el registro el actual, se produce el evento **Reposition** después de que cada registro se convierta en el actual.

Este evento se puede usar para realizar cálculos basándose en los datos del registro actual o para cambiar el formulario en respuesta a los datos del registro actual.

Validate

Se produce antes de que un registro diferente se convierta en el registro actual; antes del método Update (excepto cuando los datos se guardan con el método UpdateRecord); y de los métodos Delete, Unload o la operación Close.

Private Sub objeto_**Validate** ([índice As Integer,] acción As Integer, guardar As Integer)

objeto = Nombre del control Data

índice = Indice del control Data dentro de una matriz de controles (Si ha lugar)
acción = Un entero que indica la operación que ha producido el evento, como se describe mas adelante
guardar = una expresión booleana que especifica si los datos asociados han cambiado, como se describe mas adelante.

Acción puede tomar estos valores:

0	vbDataActionCancel	Cancela la operación cuando se sale del procedimiento Sub.
1	vbDataActionMoveFirst	Método MoveFirst.
2	vbDataActionMovePrevious	Método MovePrevious.
3	vbDataActionMoveNext	Método MoveNext.
4	vbDataActionMoveLast	Método MoveLast.
5	vbDataActionAddNew	Método AddNew.
6	vbDataActionUpdate	Operación Update (no UpdateRecord).
7	vbDataActionDelete	Método Delete.
8	vbDataActionFind	Método Find.
9	vbDataActionBookmark	La propiedad Bookmark no ha sido definida.
10	vbDataActionClose	Método Close.
11	vbDataActionUnload	El formulario se va a descargar.

Los valores de guardar son:

True	Los datos asociados han cambiado.
False	Los datos asociados no han cambiado.

El evento **Validate** se usa para realizar las últimas comprobaciones sobre los registros que se van a escribir en la base de datos.

Vea la ayuda de Visual Basic para mayor detalle de este evento.

Para los demás eventos, es aplicable lo mismo que para el resto de los controles.

CONTROLES ENLAZADOS A DATOS

Los controles enlazados a datos permiten crear aplicaciones con acceso a datos con muy poco código, o incluso ninguno. Para utilizar cualquiera de estos controles enlazados conectables a datos debe incluir uno o más controles **Data** en un formulario. El control Data establece un enlace entre la base de datos y los controles enlazados para la manipulación de los datos. El control **Data** que sirve de enlace entre la Base de Datos y los controles enlazados debe estar obligatoriamente en el mismo formulario que los controles.

Existen en Visual Basic diez controles enlazados a datos, además del control Data:

Data Ya comentado, se utiliza para tener acceso a los datos de las bases a través de controles enlazados de un formulario. Crea y administra los objetos **Database** y **Recordset** para su uso por parte de los controles enlazados. Requerido para su uso con todos los demás controles enlazados.

DBCombo Se utiliza para obtener una combinación enlazada formada por un cuadro de lista y un cuadro de texto. La lista puede llenarse automáticamente a partir de un control Data. El usuario puede elegir un elemento de la lista o introducir un valor en el cuadro de texto. Puede utilizarse para proporcionar acceso de lectura / escritura a un campo de texto específico seleccionado en la lista.

DBList Se usa para mostrar una lista generada a partir de un control Data en la que el usuario puede elegir un elemento. La lista puede rellenarse

automáticamente desde un control Data, y puede proporcionar acceso de lectura / escritura a un campo de texto específico seleccionado en ella.

DBGrid Se utiliza para obtener una combinación enlazada formada por varios registros. La cuadrícula puede rellenarse automáticamente a partir de un control Data. El usuario puede elegir un elemento de la cuadrícula o introducir un valor en un nuevo registro. Puede utilizarse para proporcionar acceso de lectura / escritura a un objeto Recordset, y para mover el registro actual del objeto Recordset asociado al control Data.

Label Se usa para el texto que el usuario no debe modificar. Puede utilizarse para ofrecer acceso de sólo lectura a un campo de texto específico.

TextBox Se utiliza para almacenar texto que el usuario puede introducir o modificar. Puede utilizarse para proporcionar acceso de lectura / escritura a un campo de texto específico.

CheckBox Se utiliza para crear un cuadro que el usuario puede elegir de forma sencilla para indicar si algo es verdadero o falso, o para mostrar varias opciones entre las que el usuario pueda elegir más de una. Puede utilizarse para proporcionar acceso de lectura / escritura a un campo booleano o de bit específico.

ComboBox Se utiliza para obtener una combinación de un cuadro de lista y un cuadro de texto. La lista se rellena con el método **AddItem**. El usuario puede elegir un elemento de la lista o introducir un valor en el cuadro de texto. Puede utilizarse para proporcionar acceso de lectura / escritura a un campo de texto seleccionado en la lista. Consulte el control DBCombo

ListBox Se utiliza para mostrar una lista en la cual el usuario puede elegir un elemento. La lista se rellena con el método **AddItem**. Puede usarse para

proporcionar acceso de lectura / escritura a un campo de texto específico seleccionado en la lista. Consulte el control DBList

PictureBox Se usa para mostrar una imagen gráfica de un mapa de bits, un icono o un meta-archivo en un formulario. Puede utilizarse para proporcionar acceso de lectura / escritura a un campo de imagen o binario específico.

Image Se utiliza para mostrar una imagen gráfica de un mapa de bits, un icono o un meta-archivo en un formulario. Las imágenes mostradas en un control Image utilizan menos recursos que las de los controles PictureBox. Puede usarse para proporcionar acceso de lectura / escritura a un campo de imagen o binario específico.

PROPIEDADES RELACIONADAS CON DATOS COMUNES A ESTOS CONTROLES

Los controles enlazados a datos tienen unas propiedades para el acceso a datos basadas en el enlace con la base de datos a través del control Data. Las propiedades comunes a todos ellos son :

DataSource Fuente de datos. Es el nombre del control **Data** que lo enlaza con la B.D. Este control Data es el que determina la Tabla donde están los campos con los datos. Esta Tabla se determina mediante la propiedad RecordSource del control Data.

DataField Es el nombre del campo, dentro de la Tabla de la base de datos, que se va a presentar en el control enlazado a datos. Esta propiedad no la tiene el **DBGrid** debido a que presenta todos los campos de la Tabla de la base de datos seleccionada en el control Data. Si se le ha forzado al control Data la propiedad Recordset, los campos que se pueden mostrar en los controles enlazados a datos son justamente, los de ese Recordset.

Cada uno de los controles enlazados a datos tiene su aplicación específica dentro del proceso de presentación y escritura de una Base de Datos.

Control Label

Posiblemente el control Label es el control más sencillo para mostrar el contenido de un campo de una base de datos. Como todos los controles enlazados a datos, el Label permite presentar los datos e introducirlos en la base a través del control Data. Lo que ocurre con el Label es que su propiedad Caption no se puede introducir directamente por teclado, y deberá cambiarse por código. Esto puede ser una ventaja (no hay posibilidad de introducirlo accidentalmente) y un inconveniente, al tener que escribir código para hacerlo.

El control Label, al poder ser origen (a través del formulario que lo contiene) y destino de un enlace DDE, esto nos puede resolver muchos problemas de introdución de distintos datos de otras aplicaciones que no tengan acceso directo a una base de datos.

El control Label, en lo referente al enlace a datos, solamente tiene las propiedades mencionadas de **DataSource** y **DataField**.

Control TextBox

Todo lo dicho del control Label es aplicable al TextBox, que además presenta la particularidad de que en este control sí se puede escribir directamente desde el teclado.

Control CheckBox

El control CheckBox permite presentar e introducir datos de tipo Booleano. Tiene

las propiedades **DataSource** y **DataFieId** en lo relativo a acceso a datos.

Controles ListBox y **ComboBox**

Estos controles tienen una característica especial respecto a su comportamiento con el enlace a la base de datos. La lista no se puede cargar directamente desde la base de datos, sino a través de un control intermedio, por ejemplo un Label, donde presentaremos un campo de la base de datos. El texto de la propiedad Caption de este Label se introduce en el ListBox o ComboBox mediante el método AddItem. Una vez introducidos todos los elementos de ese campo que nos interesen, cada vez que la base de datos se sitúa sobre el registro correspondiente a uno de los elementos que está en el ListBox o ComboBox, éste cambia su ListIndex para seleccionar el elemento correspondiente al registro actual del control Data.

El ListBox puede contener elementos correspondientes a la base de datos y otros ajenos. Puede emplearse esta característica del ListBox para seleccionar un elemento entre varios elementos tomados de la BD y otros introducidos por otro procedimiento, con la particularidad de que el ListIndex de este ListBox irá a posicionarse sobre el elemento de la BD correspondiente al registro actual del Control Data.

Tiene las propiedades **DataSource** y **DataField** en lo relativo a acceso a datos.

Control PictureBox y **Control Image**

Pueden mostrar una imagen almacenada en una Base de Datos. Mediante el control Data, se puede introducir la imagen presente en uno de estos controles en la Base de Datos.

El campo que contenga una imagen en una BD debe ser **BINARIO LARGO**, y el

tipo de imágenes que se pueden introducir son los mapas de bits (Archivos con extensión .BMP), los archivos de icono, (Extensión .ICO) y los metaarchivos. (Metafiles, extensión .WMF)

Para introducir un gráfico en una BD es mas práctico introducirlo mediante un control Data y un control Picture o Image que creando por código un Recordset. En realidad deberíamos decir que es el único método práctico de introducir / sacar imágenes de una Base de Datos

Control DBGrid

Es posiblemente el control que mas se use para presentar y modificar datos de una B.D. El control **DBGrid** presenta todos los registros y todos los campos de la Base de Datos. Por eso, necesita obligatoriamente un control Data para poder presentar datos. Otros controles (Label, TextBox, Picture, etc.) que solamente presentan un dato (un campo de un registro) pueden trabajar sin necesidad de un control data, creando un Recordset mediante código. (Lo veremos un poco mas adelante). Sin embargo el control DBGrid, al presentar todos los datos de la base de datos necesita un control Data. Veamos porqué.

Cuando creamos un objeto Recordset mediante la instrucción: (se verá mas adelante)

Set Mirecordset = MiDataBase.**Openrecordset** ("Select campo1, campo2 from Mitabla")

lo que estamos haciendo es seleccionar, de todos los campos que pueda tener la tabla llamada Mitabla, los denominados campo1 y campo2. Cada vez que seleccionemos un registro, es ese registro solamente el que se mantiene en la memoria del ordenador, (el registro actual) y de ese registro, solamente metemos los datos del campo1 y campo2.

Cuando creamos un Recordset mediante un control Data, se meten en la memoria TODOS los registros de la tabla especificada en el control data. Por lo tanto, al permanecer todos los registros de esa tabla en memoria, podremos presentar sus valores en el control **DBGrid**. No lo podremos hacer con un Recordset creado mediante código, que solamente mantiene un registro en memoria.

Deberemos explicar qué ocurre cuando se crea un Recordset mediante código, y posteriormente se fuerza a que el Recordset del control data sea igual a ese Recordset creado.

Con el Recordset creado con la instrucción anterior, podemos forzar a un control Data que su Recordset sea igual al ya creado mediante la instrucción:

Set Data1.**Recordset**=MiRecordset

En este caso, el control Data1 tomará todos los registros con los campos campo1 y campo2 de la base de datos y los meterá en la memoria RAM. Así ya podemos rellenar las cuadrículas del control DBGrid.

Observe que una aplicación de acceso a datos ocupará mucha mas memoria RAM si establecemos el enlace con la base de datos mediante un control Data que si lo hacemos creando Recordsets a medida. Pero si necesitamos presentar los datos en un DBGrid, no quedará mas remedio que usar un control Data. Si nuestra aplicación no tiene que presentar en el DBGrid todos los campos de la tabla de la B.D. podemos crear previamente un Recordset mediante código y a continuación forzar que el Recordset del control Data sea igual al Recordset creado, utilizando la expresión anterior.

El control DBGrid tendrá tantas columnas como campos tenga el Recordset. El número de filas será igual al número de registros que tiene la tabla. Si se

sobrepasa el espacio físico del DBGrid para poder presentarlos, aparecerán automáticamente flechas de deslizamiento vertical. El ancho de las columnas puede cambiarse mediante la propiedad **Width** del objeto **Columns** del DBGrid.

DBGrid1.Columns(n).**Width** = Valor

Donde n es el número de la columna (la primera es la 0) y el valor debe expresarse según las unidades de medida (ScaleWidth) del Formulario que lo contiene.

Del control DBGrid podemos destacar estas propiedades:

AllowAddNew

Devuelve o establece un valor que indica si el usuario puede agregar nuevos registros al objeto Recordset subyacente a un control DBGrid.

La última fila que se muestra en el control DBGrid se deja en blanco para permitir a los usuarios introducir nuevos registros. Si la propiedad **AllowAddNew** es False, los usuarios no pueden establecer el foco en dicha fila.
El Recordset subyacente puede, por otras razones, no permitir inserciones incluso en el caso de que la propiedad AllowAddNew sea True. En este caso, se producirá un error si el usuario intenta agregar un registro.

AllowDelete

Devuelve o establece un valor que indica si el usuario puede **eliminar registros** del objeto Recordset subyacente a un control DBGrid.

Utilice la propiedad **AllowDelete** para impedir que los usuarios eliminen registros del conjunto de registros a través de la interacción con el control DBGrid.

El objeto Recordset subyacente puede, por otras razones, no permitir eliminaciones incluso en el caso de que la propiedad AllowDelete sea True. En este caso, se producirá un error si el usuario intenta eliminar un registro.

AllowRowSizing

Devuelve o establece un valor que indica si un usuario puede modificar el tamaño de las filas del control DBGrid.

Sintaxis nombre.**AllowRowSizing** = [True / False]

Si la propiedad AllowSizing es True, el puntero del mouse se convierte en una flecha de doble cabeza (Size N S) cuando se sitúa sobre el divisor de filas entre selectores de registro, y el usuario puede modificar el tamaño de las filas mediante arrastre. Cualquier cambio de tamaño de columna provoca un evento RowResize.

AllowUpdate

Devuelve o establece un valor que indica si un usuario puede modificar datos del control DBGrid.

Sintaxis nombre.**AllowUpdate** = [True / False]

Cuando la propiedad **AllowUpdate** es False, el usuario puede aún desplazarse a través del control DBGrid y seleccionar datos, pero no puede modificar ninguno de los valores; cualquier intento de hacerlo se ignora.
Puede también hacer uso de las propiedades del objeto **Columns** para hacer que columnas individuales del control DBGrid sean de sólo lectura, pero los valores de la propiedad AllowUpdate tienen prioridad sobre los valores establecidos para las columnas (sin modificar éstos).

Nota El objeto Recordset puede no permitir actualizaciones incluso si AllowUpdate es True para el control DBGrid; en este caso se produce un error interceptable cuando el usuario intenta cambiar el registro.

ColumnHeaders

Devuelve o establece un valor que indica si los encabezados de columna se muestran en el control DBGrid.

Sintaxis objeto.**ColumnHeaders** = [True / False]

Si es True se muestran los encabezados de columna del control DBGrid, y si es False no se muestran.

DataMode

Establece un valor que especifica si el control DBGrid funciona en modo enlazado o no enlazado. Esta propiedad no está disponible en tiempo de ejecución.

Los valores que puede tomar la propiedad DataMode son:

0-Bound. El control DBGrid está enlazado con el control Data.
1-Unbound. El control DBGrid no está enlazado directamente al control Data.

Un DBGrid está enlazado cuando se le asigna un control Data en su propiedad DataSource. En este caso, presenta sin mas los datos del Recordset de ese control Data. Si le especificamos en la propiedad DataMode que no esté enlazado, utilizaremos código en los procedimientos del control Data para pasarle los datos cuando nos interese.

DefColWidth

Devuelve o establece un valor que indica el ancho de columna predeterminado para todas las columnas del control DBGrid.

Sintaxis objeto.**DefColWidth** [= valor]

donde valor es un entero basado en el modo de escala del control.

Si se da a la propiedad DefColWidth el valor 0, el control establece automáticamente el tamaño de todas las columnas en base al ancho del encabezado de columna o al valor de la propiedad Size del campo subyacente, seleccionando el más largo de los dos.

RecordSelectors

Los selectores de registros aparecen a la izquierda de las filas en el control DBGrid. Cuando el usuario elige el selector, el registro completo (fila del control DBGrid) se selecciona.

La propiedad **RecordSelectors** devuelve o establece un valor que indica si se muestran los selectores de registro en el control DBGrid.

Sintaxis objeto.**RecordSelectors** = [True / False]

El Objeto Columns aplicado al control DBGrid.

El objeto **Columns** es un objeto no privativo del control DBGrid, que contiene todas las columnas y las propiedades de las columnas de un control. Podemos cambiar las propiedades de cada una de las columnas de un DBGrid mediante las

propiedades de los objetos Columns asociado a él. Por ejemplo, el encabezamiento de una columna en un DBGrid es, por defecto, el nombre del campo que se va a presentar en esa columna. Si queremos poner otro encabezamiento a una columna, ejecutaremos la expresión :

DBGrid1.**Columns(**0**)**.Caption = "Cabecera"

donde el 0 entre paréntesis significa que estamos afectando a la columna número 0 (la primera por la izquierda).

Si queremos cambiar su anchura:

DBGrid1.**Columns(**3**)**.Width = 1000

En este caso estamos fijando la anchura de la columna cuarta por la izquierda a 1000 unidades de medida de las del Formulario que contiene al DBGrid.

CONTROLES DBList y DBCombo

Los dos controles **DBList** y **DBCombo** se implementan de la misma manera. Las dos únicas diferencias estriban en la forma en que se presenta la información al usuario y la presencia de la porción del control **DBCombo** en el cuadro de texto, que se emplea para introducir valores.

Los controles **DBList** y **DBCombo** tienen dos modos que pueden utilizarse individualmente o al mismo tiempo:

Autollenado: Llena automáticamente la lista con un campo seleccionado de entre todos los registros administrados por el control Data especificado por la propiedad

RowSource del control DBList o DBCombo.

Actualización automática: Enlaza el registro seleccionado en el control a un campo específico del objeto Recordset administrado por el control Data especificado por la propiedad **DataSource**.

Esto explicado en otras palabras significa lo siguiente:

En control DBList o DBCombo puede trabajar sobre dos controles Data. Uno para rellenar la lista. El control Data y el campo que rellena la lista son los especificados en las propiedades **RowSource** y **ListField** del control DBList o DBCombo. Respecto a este control Data estos controles funcionan solamente como receptores de datos : No pueden cambiar el contenido de los registros con los que rellenan su lista. (Llamemos a esta base de datos Base A en esta explicación)

El otro control Data es el que estos controles usan para introducir datos en su BD asociada. El control Data y el campo de la BD asociados a estos controles DBList y DBCombo, son los especificados en las propiedades **DataSource** y **DataField**. Es sobre esta base de datos y el campo correspondiente sobre los que estos controles DBList y DBCombo actúan cambiando o introduciendo datos. (Llamemos a esta otra base de datos Base B)

Basta con seleccionar un elemento de la lista (que pertenece a la base A) y ese elemento se colocará en el campo correspondiente (el indicado en la propiedad **DataField** del control DBList o DBCombo) de la base B. En el caso del DBCombo, podemos escribir directamente el dato en su caja de texto en vez de seleccionarlo de la lista.

No tiene porqué ser el elemento que vemos en la lista el que pasemos a la otra base de datos. Puede ser otro campo del mismo Recordset. Depende de la propiedad **BoundColumn** que viene a continuación. Léasela con la atención que

se merece.

Vemos pues que estos controles pueden servir como un puente entre dos bases de datos.

Veamos unas propiedades especificas de estos dos controles:

BoundColumn

Devuelve o establece el nombre del campo de origen de un objeto Recordset que se utiliza para suministrar un valor de datos a otro Recordset, o el nombre del campo empleado para rellenar un control DBList o DBCombo.

Resumamos. En un DBList o DBCombo presentamos en su lista un determinado campo de una BD. Esa BD tendrá más campos. Un poco mas arriba decíamos que ese elemento de la lista podíamos pasarlo a otra BD (La especificada en la propiedad **DataSource**, y en su campo **DataField**). ¿Podríamos pasar a esa BD, en vez del elemento de la lista, otro campo de esa BD origen ?. Podríamos, por ejemplo, presentar en la lista el nombre de una persona, nombre que hemos tomado de un listín telefónico, y en vez de pasar el nombre que es el que figura en la lista, pasar su número de teléfono, que es otro campo de la misma BD. La respuesta es SI. Para ello, pongamos en la propiedad **BoundColumn** del DBList o DBCombo que estamos usando, el nombre del campo que contiene el número de teléfono. Observe que por defecto, esa propiedad se rellena con el mismo campo que el especificado en la propiedad **ListField**. Eso no quiere decir que no se pueda cambiar. Para cambiarlo, haga click en la flecha vertical que aparece en la casilla de propiedades, y donde verá que aparecen todos los campos de la base de datos seleccionada en el control Data asociado a este control. También puede cambiarlo en tiempo de ejecución con la siguiente sintaxis:

nombredelDBList.**BoundColumn** = nombredelcampo

Con estas ideas expresadas aquí, puede comenzar a leer el texto de ayuda de esta propiedad. No se desespere si no entiende algo de lo allí expresado.

BoundText

Devuelve o establece el valor de la propiedad BoundColumn de un control DBCombo o DBList pasado desde o hacia la propiedad DataField después de realizar una selección. Es decir, es el contenido del campo especificado en la propiedad **BoundColumn** comentada anteriormente.

Esta propiedad está disponible solamente en tiempo de ejecución.

Esta propiedad es de lectura y escritura. Es sencilla de usar en para conocer el contenido del campo especificado en **BoundColumn**. (Lectura del valor)

Cuando la queramos utilizar para forzar el valor de esta propiedad a un valor determinado, debemos utilizar la siguiente sintaxis:

objeto.**BoundText** [= valor]

En este caso, el DBList o DBCombo intenta buscar un elemento coincidente en el campo especificado en la propiedad **BoundColumn** de todos los registros de la BD asociada. Si encuentra uno igual, se establece el valor de la propiedad **BoundText** basándose en el campo especificado por la propiedad **BoundColumn**. Si no se encuentra dicha coincidencia, la propiedad **BoundText** se establece en el valor Null.

Es decir, si hay coincidencia con algún valor de ese campo, BoundText seguirá con el valor especificado. Si no la hay, **BoundText** se pone a Nulo.

MatchWithList

Propiedad solo de lectura. Devuelve True si el contenido actual de la propiedad BoundText coincide con uno de los registros de la parte de lista del control.

Sintaxis Variable = objeto.**MatchWithList**

Si Variable = True el contenido de la propiedad BoundText coincide con uno de los registros de la lista. Si es False, el contenido de la propiedad BoundText no coincide con ninguno de los registros de la lista.

Cuando introduce un valor en la parte de texto del control DBCombo, la propiedad MatchWithList se establece como True si el valor introducido es uno de los elementos que aparecen en la lista. Usando esta propiedad, el código puede interceptar entradas que no están la lista, o proporcionar código para agregar la nueva entrada a la tabla de origen.

DataChanged

Devuelve o establece un valor que indica que han cambiado los datos del control enlazado por algún proceso distinto de la recuperación de datos del registro actual. No está disponible en tiempo de diseño.

Sintaxis objeto.**DataChanged** [=Variable] ‘establece la propiedad
Variable = objeto.**DataChanged** ‘lee el valor actual de esta propiedad

Variable puede ser True, indicando que los datos que hay actualmente en el control no son iguales que los del registro actual, y False (Predeterminado) que indica que los datos que hay actualmente en el control (si los hay) son iguales que

los del registro actual.

Comentarios

Cuando un control Data se mueve de un registro a otro, pasa datos desde los campos del registro actual a controles enlazados al campo específico o el registro completo. Cuando se muestran datos en los controles enlazados, la propiedad **DataChanged** se establece como False. Si el usuario o alguna operación cambian el valor del control enlazado, la propiedad DataChanged se establece como True. Si pasa a otro registro la propiedad DataChanged no se ve afectada.
Cuando el control Data comienza a mover a otro registro, se produce el evento Validate. Si DataChanged es True para algún control enlazado, el control Data invoca automáticamente los métodos Edit y Update para enviar los cambios a la base de datos.

Si no desea guardar los cambios de un control enlazado en la base de datos, puede establecer la propiedad DataChanged como False en el evento Validate.

MatchEntry

Devuelve o establece un valor que indica cómo el control DBCombo o DBList realiza búsquedas basándose en la entrada del usuario.

Sintaxis objeto.**MatchEntry** [= valor]

Donde valor es una constante o un valor que define el comportamiento de un control cuando tiene el enfoque y el usuario introduce uno o más caracteres.

0 vbMatchEntrySimple Coincidencia básica: (Predeterminado) El control busca la siguiente coincidencia del carácter introducido usando la primera letra de

entradas de la lista. Al escribir repetidamente la misma letra se recorren todas las entradas de la lista que comienzan por esa letra.

1 vbMatchEntryExtended Coincidencia ampliada: El control busca una entrada que coincida con todos los caracteres introducidos. La búsqueda se realiza a medida que se escriben los caracteres, refinando progresivamente la búsqueda.

Cuando la propiedad MatchEntry se establece como vbMatchEntryExtended y el usuario presiona la tecla de retroceso o espera varios segundos, la cadena de coincidencias de restablece.

SelectedItem (Solo DBCombo)

Devuelve un valor que contiene un marcador para el registro seleccionado en un control DBCombo.

Sintaxis DBCombo1.**SelectedItem**

Cuando selecciona un elemento de la parte de lista del control DBCombo, la propiedad SelectedItem contiene un marcador que puede utilizar para reposicionar el registro seleccionado en el Recordset como especifica la propiedad RowSource.

SelText

SelText devuelve o establece una cadena con el texto actualmente seleccionado, o es una cadena de longitud cero () si no hay caracteres seleccionados.

VisibleCount

Devuelve un valor que indica el número de elementos visibles del control DBCombo o DBList.

Sintaxis objeto.**VisibleCount**

La propiedad VisibleCount devuelve un entero desde 0 al número de elementos visibles del control. Un elemento se considera visible únicamente si una parte del texto es visible.

VisibleItems

Devuelve una matriz marcadores, uno para cada elemento visible de la lista del control DBCombo o DBList.

Sintaxis objeto.**VisibleItems**

Estos marcadores pueden emplearse para obtener registros individuales del conjunto de registros empleado para rellenar la lista.